AMD-3D™

Library Reference Guide

Contents

1. Introduction to AMD-3D Library

The goal of this library is to provide examples and working functions which show off the capabilities of the AMD-3D hardware. The goal is not to provide a full 3D or DSP library. As such the focus is on floating point intensive operations which occur in various categories. These categories range from 3D graphics to DSP to Image processing, etc. The library is provided as a ASM only interface using MASM or, with slight modifications, other x86 assemblers and assuming a flat memory model. Full source code is provided for any desired customization or to use as examples for custom AMD-3D development. Parameters are passed in registers in most cases, for performance reasons. The library passes pointers to structures in non-MMX registers. Very few return values are given as the AMD-3D operations most always produce no exceptions. In cases where there are scalar return values, they are usually returned in the MM0 register. The source code requires use of MASM 6.13 beta or MASM 5.1 or higher and the included AMD-3D macros. For sample C code and the inline assembly macros you must have MSVC++ 4.2 or higher.(use the /GM switch for V 4.1 and 4.2)

2. AMD-3D Library Overview

This section overviews the function categories, data structures, conventions and reference format.

2.1 Function categories

- Initialization
- Vector/Matrix Algebra
- Transcendentals
- Image processing
- Digital signal processing
- Macros
- x87 replacements

2.2 Data structures

2.2.1 Basic types

```
The basic data type for AMD 3D code is the IEEE single
precision floating point data type.

float   TYPEDEF  REAL4  ; basic data type
```

2.2.2 Point

```
point_3d STRUCT        ; 3D point structure
    x   float   ?
    y   float   ?
    z   float   ?
```

```
point_3d ENDS
; Pointer to point_3d struct
point_3d_ptr TYPEDEF PTR point_3d
```

2.2.3 Vector

```
vector_3d STRUCT       ; 3D vector structure
    x   float   ?
    y   float   ?
    z   float   ?
vector_3d ENDS
; Pointer to vector_3d struct
vector_3d_ptr TYPEDEF PTR vector_3d
```

2.2.4 Vector homogeneous

```
vector_3d_h STRUCT     ; homogeneous 3D vector structure
    x   float   ?
    y   float   ?
    z   float   ?
    w   float   ?
vector_3d_h ENDS
; Pointer to vector_3d_h struct
vector_3d_h_ptr TYPEDEF PTR vector_3d_h
```

2.2.5 Direction Vector

```
dir_vector_3d STRUCT   ; direction vector structure
    ang_x   float   ?  ; angle relative to x axis
    ang_y   float   ?  ; angle relative to y axis
    ang_z   float   ?  ; angle relative to z axis
dir_vector_3d ENDS
; Pointer to dir_vector_3d struct
dir_vector_3d_ptr TYPEDEF PTR dir_vector_3d
```

2.2.6 Matrix

```
matrix_3x3  STRUCT
    _00     float  3 DUP (?)
    _10     float  3 DUP (?)
    _20     float  3 DUP (?)
matrix_3x3  ENDS
```

2.2.7 Matrix Homogenous

```
matrix_4x4  STRUCT
    _00     float  4 DUP (?)
    _10     float  4 DUP (?)
    _20     float  4 DUP (?)
    _30     float  4 DUP (?)
matrix_4x4  ENDS
```

2.3 Coding conventions

Overview the format of code, show samples, etc.
Discussion of performance issues with development using AMD-3D.
To be added at a later date.

2.4 Naming conventions

All AMD-3D specific functions and macros are prefaced with AMD3D_ or _AMD3D_ as a means of delineating them.

2.5 Reference format

Each function is fully described on a separate page. Included is the name of the function, a synopsis which shows the form of the function call, a description of the input and output parameters, a discussion section which includes any pertinent details, and an example of the use of the function. Below is a sample of a typical page from the reference section.

Name
AMD3D_Vector_Transform - transform 1x4 vector using 4x4 matrix
Synopsis

```
lea  eax, DWORD PTR res
lea  edx, DWORD PTR mh1
lea  ecx, DWORD PTR vh1
call AMD3D_Vector_Transform
```

Description
Input Parameters
mh1 - a pointer to a 4x4 dimensional homogeneous matrix which is the transformation matrix. Passed in the ECX register.
vh1 - a pointer to a 1x4 dimensional homogeneous matrix holding the x, y, z and w values. Passed in the EDX register.
Output Parameters
res - a pointer to a 1x4 dimensional homogeneous matrix holding the result of the vector transform operation. Passed in the EAX register.
Discussion
This function performs a matrix multiplication of a vector_3d_h type with a matrix_4x4 type producing a resultant vector_3d_h value.
Example

```
vh1   vector_3d  {1.0, 2.0, 3.0, 1.0}
mh1    matrix_4x4 {{5.0, 2.0, 4.0, 6.0},
                   {1.0, 4.0, 7.0, 2.0},
                   {2.0, 8.0, 3.0, 1.0},
                   {0.0, 0.0, 0.0, 1.0}}
vh1new vector_3d ?

lea  eax, DWORD PTR vh1new
lea  edx, DWORD PTR mh1
lea  ecx, DWORD PTR vh1
```

```
call  AMD3D_Vector_Transform

push  DWORD PTR vh1new
call  print_matrix_4x4

x = 13, y = 34, z = 27, w = 14
```

3. AMD-3D Initialization

CPU detection and enabling of processor specific capabilities

Name

AMD3D_Detect_Hardware - determine if AMD-3D aware Hardware is present.

Synopsis

```
call AMD3D_Detect_Hardware
```

Description

Input Parameters

None

Output Parameters

EAX - 0 = AMD-3D hardware is present.

1 = AMD-3D hardware is not present

Discussion

This function uses the CPUID instruction to determine if AMD-3D capable Hardware is present.

Example

```
call  AMD3D_Detect_Hardware

push  eax
call  print_int
```

4. Vector Algebra

This section describes the various vector and matrix operations supported.

4.1 Vector/Vector add

Name

AMD3D_Vector_Add - add 2 1x4 vectors

Synopsis

```
lea  eax, DWORD PTR result
lea  edx, DWORD PTR vh1
lea  ecx, DWORD PTR vh2
call AMD3D_Vector_Add
```

Description

Input Parameters

vh1 - a pointer to a 1x4 dimensional homogeneous matrix holding the x, y, z and w values. Passed in the EDX register.

vh2 - a pointer to a 1x4 dimensional homogeneous matrix holding the x, y, z and w values. Passed in the ECX register.

Output Parameters

result - a pointer to a 1x4 dimensional homogeneous matrix holding the result of the vector add operation. Passed in the EAX register.

Discussion

This function performs a matrix addition where both matrices are vector_3d type. The result is also a vector_3d type.

Example

```
vh1    vector_3d {1.0, 2.0, 3.0, 1.0}
vh2    vector_3d {2.0, 4.0, 5.0, 1.0}
vh1new vector_3d ?

lea  eax, DWORD PTR vh1new
lea  edx, DWORD PTR vh1
lea  ecx, DWORD PTR vh2
call  AMD3D_Vector_Add

push  DWORD PTR vh1new
call  print_vector
```

[show results]

4.2 Vector/Scalar add

Name

AMD3D_Vector_Scalar_Add - add a scalar to a 1x4 vector

Synopsis

```
lea  edx, DWORD PTR result
lea  ecx, DWORD PTR vh1
movd  mm0, k
call AMD3D_Vector_Scalar_Add
```

Description

Input Parameters

vh1 - a pointer to a 1x4 dimensional homogeneous matrix holding the x, y, z and w values. Passed in the ECX register.

k - the scalar constant. Passed in the mm0 register.

Output Parameters

result - a pointer to a 1x4 dimensional homogeneous matrix holding the result of the vector scalar add operation. Passed in the EDX register.

Discussion

This function performs a scalar addition to a matrix where the matrix is a vector_3d type. The scalar is floating point type. The result is a vector_3d type.

Example

```
vh1    vector_3d {1.0, 2.0, 3.0, 1.0}
scalar_k  float   10.0
vh1new vector_3d  ?

lea  edx, DWORD PTR vh1new
lea  ecx, DWORD PTR vh1
movd scalar_k
call  AMD3D_Vector_Scalar_Add

push  DWORD PTR vh1new
call  print_vector
```

[show results]

4.3 Vector/Vector sub

Name

AMD3D_Vector_Sub - subtract 2 1x4 vectors

Synopsis

```
lea  eax, DWORD PTR result
lea  edx, DWORD PTR vh1
lea  ecx, DWORD PTR vh2
call AMD3D_Vector_Sub
```

Description

Input Parameters

vh1 - a pointer to a 1x4 dimensional homogeneous matrix holding the x, y, z and w values. Passed in the EDX register.

vh2 - a pointer to a 1x4 dimensional homogeneous matrix holding the x, y, z and w values. Passed in the ECX register.

Output Parameters

result - a pointer to a 1x4 dimensional homogeneous matrix holding the result of the vector subtraction operation(vh1 - vh2). Passed in the EAX register.

Discussion

This function performs a matrix subtraction where both matrices are vector_3d type. The result is also a vector_3d type.

Example

```
vh1    vector_3d {1.0, 2.0, 3.0, 1.0}
vh2    vector_3d {2.0, 4.0, 5.0, 1.0}
vh1new vector_3d ?

lea  eax, DWORD PTR vh1new
lea  edx, DWORD PTR vh1
lea  ecx, DWORD PTR vh2
call  AMD3D_Vector_Sub

push  DWORD PTR vh1new
call  print_vector
```

[show results]

4.4 Vector/Scalar sub

Name

AMD3D_Vector_Scalar_Sub - subtract a scalar from a 1x4 vector

Synopsis

```
lea  edx, DWORD PTR result
lea  ecx, DWORD PTR vh1
movd  mm0, k
call AMD3D_Vector_Scalar_Sub
```

Description

Input Parameters

vh1 - a pointer to a 1x4 dimensional homogeneous matrix holding the x, y, z and w values. Passed in the ECX register.

k - the scalar constant. Passed in the mm0 register.

Output Parameters

result - a pointer to a 1x4 dimensional homogeneous matrix holding the result of the vector scalar add operation. Passed in the EDX register.

Discussion

This function performs a scalar subtraction from a matrix where the matrix is a vector_3d type. The scalar is floating point type. The result is a vector_3d type.

Example

```
vh1    vector_3d {1.0, 2.0, 3.0, 1.0}
scalar_k  float   10.0
vh1new vector_3d  ?

lea  edx, DWORD PTR vh1new
lea  ecx, DWORD PTR vh1
movd scalar_k
call  AMD3D_Vector_Scalar_Sub

push  DWORD PTR vh1new
call  print_vector
```

[show results]

4.5 Vector/Vector mul

Name

AMD3D_Vector_Mul - multiply 2 1x4 vectors

Synopsis

```
lea  eax, DWORD PTR result
lea  edx, DWORD PTR vh1
lea  ecx, DWORD PTR vh2
call AMD3D_Vector_Mul
```

Description

Input Parameters

vh1 - a pointer to a 1x4 dimensional homogeneous matrix holding the x, y, z and w values. Passed in the EDX register.

vh2 - a pointer to a 1x4 dimensional homogeneous matrix holding the x, y, z and w values. Passed in the ECX register.

Output Parameters

result - a pointer to a 1x4 dimensional homogeneous matrix holding the result of the vector multiply operation. Passed in the EAX register.

Discussion

This function performs a simple vector multiplication where the corresponding values are multiplied together. Both matrices are vector_3d type. The result is also a vector_3d type.

Example

```
vh1    vector_3d {1.0, 2.0, 3.0, 1.0}
vh2    vector_3d {2.0, 4.0, 5.0, 1.0}
vh1new vector_3d ?

lea  eax, DWORD PTR vh1new
lea  edx, DWORD PTR vh1
lea  ecx, DWORD PTR vh2
call  AMD3D_Vector_Mul

push  DWORD PTR vh1new
call  print_vector
```

[show results]

4.6 Vector/Scalar mul

Name

AMD3D_Vector_Scalar_Mul - multiply a scalar with a 1x4 vector

Synopsis

```
lea  edx, DWORD PTR result
lea  ecx, DWORD PTR vh1
movd  mm0, k
call AMD3D_Vector_Scalar_Mul
```

Description

Input Parameters

vh1 - a pointer to a 1x4 dimensional homogeneous matrix holding the x, y, z and w values. Passed in the ECX register.

k - the scalar constant. Passed in the mm0 register.

Output Parameters

result - a pointer to a 1x4 dimensional homogeneous matrix holding the result of the vector scalar multiply operation. Passed in the EDX register.

Discussion

This function performs a scalar multiplication with a matrix where the matrix is a vector_3d type. The scalar is floating point type. The result is a vector_3d type.

Example

```
vh1    vector_3d {1.0, 2.0, 3.0, 1.0}
scalar_k  float   10.0
vh1new vector_3d  ?

lea  edx, DWORD PTR vh1new
lea  ecx, DWORD PTR vh1
movd scalar_k
call  AMD3D_Vector_Scalar_Mul

push  DWORD PTR vh1new
call  print_vector
```

[show results]

4.7 Vector normalize

Name

AMD3D_Vector_Normalize - Normalize a 1x4 vector

Synopsis

```
lea  ecx, DWORD PTR v1
lea  edx, DWORD PTR vh1
call AMD3D_Vector_Normalize
```

Description

Input Parameters

vh1 - a pointer to a 1x4 dimensional homogeneous matrix holding the x, y, z and w values. Passed in the ECX register.

Output Parameters

result - a pointer to a 1x4 dimensional homogeneous matrix holding the result of the vector normalize operation. Passed in the EDX register.

Discussion

This function performs a vector normalization to a matrix where the matrix is a vector_3d type. The result is a vector_3d type.

Example

```
vh1    vector_3d {1.0, 2.0, 3.0, 1.0}
vh1new vector_3d  ?

lea  ecx, DWORD PTR vh1
lea  edx, DWORD PTR vh1new
call  AMD3D_Vector_Normalize

push  DWORD PTR vh1new
call  print_vector
```

[show results]

4.8 Vector magnitude

Name

AMD3D_Vector_Magnitude - Determine the Magnitude of a 1x4 vector

Synopsis

```
lea  ecx, DWORD PTR v1
lea  edx, DWORD PTR result
call AMD3D_Vector_Magnitude
```

Description

Input Parameters

vh1 - a pointer to a 1x4 dimensional homogeneous matrix holding the x, y, z and w values. Passed in the ECX register.

Output Parameters

result - a pointer to a float holding the result of the vector magnitude operation. Passed in the EDX register.

Discussion

This function performs a vector magnitude calculation to a matrix where the matrix is a vector_3d type. The result is a float type.

Example

```
vh1    vector_3d {1.0, 2.0, 3.0, 1.0}
result float     ?

lea  ecx, DWORD PTR vh1
lea  edx, DWORD PTR result
call  AMD3D_Vector_Magnitude

push  DWORD PTR result
call  print_float
```

[show results]

4.9 Matrix addition

4.9.1 Matrix Add (3x3 + 3x3)

Name

AMD3D_Matrix_Add_3x3 - add 2 3x3 matrices

Synopsis

```
lea  eax, DWORD PTR result
lea  edx, DWORD PTR m1
lea  ecx, DWORD PTR m2
call AMD3D_Matrix_Add_3x3
```

Description

Input Parameters

m1 - a pointer to a 3x3 dimensional matrix. Passed in the EDX register.

m2 - a pointer to a 3x3 dimensional matrix. Passed in the ECX register.

Output Parameters

result - a pointer to a 3x3 dimensional matrix holding the result of the matrix addition operation. Passed in the EAX register.

Discussion

This function performs a matrix addition where both matrices are 3x3 dimension. The result is also a matrix of 3x3 dimension.

Example

```
tm1    matrix_3x3 {{5.0, 2.0, 4.0},
                   {1.0, 4.0, 7.0},
                   {2.0, 3.0, 6.0}}
tm2    matrix_3x3 {{2.0, 3.0, 8.0},
                   {3.0, 3.0, 5.0},
                   {7.0, 9.0, 1.0}}
tm1new matrix_3x3  ?

lea  eax, DWORD PTR tm1new
lea  edx, DWORD PTR tm1
lea  ecx, DWORD PTR tm2
call  AMD3D_Matrix_Add_3x3

push  DWORD PTR tm1new
call  print_matrix_3x3
```

[show results]

4.9.2 Matrix Add (4x4 + 4x4)

Name

AMD3D_Matrix_Add_4x4 - add 2 4x4 matrices

Synopsis

```
lea  eax, DWORD PTR result
lea  edx, DWORD PTR m1
lea  ecx, DWORD PTR m2
call AMD3D_Matrix_Add_4x4
```

Description

Input Parameters

m1 - a pointer to a 4x4 dimensional matrix. Passed in the EDX register.

m2 - a pointer to a 4x4 dimensional matrix. Passed in the ECX register.

Output Parameters

result - a pointer to a 4x4 dimensional matrix holding the result of the matrix addition operation. Passed in the EAX register.

Discussion

This function performs a matrix addition where both matrices are 4x4 dimension. The result is also a matrix of 4x4 dimension.

Example

```
tm1    matrix_4x4 {{5.0, 2.0, 4.0, 6.0},
                   {1.0, 4.0, 7.0, 2.0},
                   {2.0, 8.0, 3.0, 1.0},
                   {4.0, 5.0, 6.0, 1.0}}
tm2    matrix_4x4 {{3.0, 1.0, 6.0, 8.0},
                   {3.0, 3.0, 8.0, 5.0},
                   {3.0, 7.0, 8.0, 9.0},
                   {0.0, 0.0, 0.0, 1.0}}
tm1new matrix_4x4  ?

lea  eax, DWORD PTR tm1new
lea  edx, DWORD PTR tm1
lea  ecx, DWORD PTR tm2
call  AMD3D_Matrix_Add_4x4

push  DWORD PTR tm1new
call  print_matrix_4x4
```

[show results]

4.10 Matrix subtraction

4.10.1 Matrix Subtraction (3x3 - 3x3)

Name

AMD3D_Matrix_Sub_3x3 - add 2 3x3 matrices

Synopsis

```
lea  eax, DWORD PTR result
lea  edx, DWORD PTR m1
lea  ecx, DWORD PTR m2
call AMD3D_Matrix_Sub_3x3
```

Description

Input Parameters

m1 - a pointer to a 3x3 dimensional matrix. Passed in the EDX register.

m2 - a pointer to a 3x3 dimensional matrix. Passed in the ECX register.

Output Parameters

result - a pointer to a 3x3 dimensional matrix holding the result of the matrix subtraction operation. Passed in the EAX register.

Discussion

This function performs a matrix subtraction (m1-m2) where both matrices are 3x3 dimension. The result is also a matrix of 3x3 dimension.

Example

```
m1    matrix_3x3  {{5.0, 2.0, 4.0},
                   {1.0, 4.0, 7.0},
                   {2.0, 3.0, 6.0}}
m2    matrix_3x3  {{2.0, 3.0, 8.0},
                   {3.0, 3.0, 5.0},
                   {7.0, 9.0, 1.0}}
m1new matrix_3x3  ?

lea  eax, DWORD PTR m1new
lea  edx, DWORD PTR m1
lea  ecx, DWORD PTR m2
call  AMD3D_Matrix_Sub_3x3

push  DWORD PTR m1new
call  print_matrix_3x3
```

[show results]

4.10.2 Matrix Subtraction (4x4 - 4x4)

Name

AMD3D_Matrix_Sub_4x4 - add 2 4x4 matrices

Synopsis

```
lea  eax, DWORD PTR result
lea  edx, DWORD PTR m1
lea  ecx, DWORD PTR m2
call AMD3D_Matrix_Sub_4x4
```

Description

Input Parameters

m1 - a pointer to a 4x4 dimensional matrix. Passed in the EDX register.

m2 - a pointer to a 4x4 dimensional matrix. Passed in the ECX register.

Output Parameters

result - a pointer to a 4x4 dimensional matrix holding the result of the matrix subtraction operation. Passed in the EAX register.

Discussion

This function performs a matrix subtraction(m1-m2) where both matrices are 4x4 dimension. The result is also a matrix of 4x4 dimension.

Example

```
m1    matrix_4x4  {{5.0, 2.0, 4.0, 6.0},
                   {1.0, 4.0, 7.0, 2.0},
                   {2.0, 8.0, 3.0, 1.0},
                   {4.0, 5.0, 6.0, 1.0}}
m2    matrix_4x4  {{3.0, 1.0, 6.0, 8.0},
                   {3.0, 3.0, 8.0, 5.0},
                   {3.0, 7.0, 8.0, 9.0},
                   {0.0, 0.0, 0.0, 1.0}}
m1new matrix_4x4  ?

lea  eax, DWORD PTR m1new
lea  edx, DWORD PTR m1
lea  ecx, DWORD PTR m2
call  AMD3D_Matrix_Add_4x4

push  DWORD PTR m1new
call  print_matrix_4x4
```

[show results]

4.11 Matrix multiplication

4.11.1 Matrix Multiplication (3x3 by 3x3)

Name

AMD3D_Matrix_Mul_3x3 - Multiplies 2 3x3 matrices

Synopsis

```
lea  eax, DWORD PTR result
lea  edx, DWORD PTR m1
lea  ecx, DWORD PTR m2
call AMD3D_Matrix_Mul_3x3
```

Description

Input Parameters

m1 - a pointer to a 3x3 dimensional matrix. Passed in the EDX register.

m2 - a pointer to a 3x3 dimensional matrix. Passed in the ECX register.

Output Parameters

result - a pointer to a 3x3 dimensional matrix holding the result of the matrix Multiplication operation. Passed in the EAX register.

Discussion

This function performs a matrix Multiplication where both matrices are 3x3 dimension. The result is also a matrix of 3x3 dimension.

Example

```
tm1    matrix_3x3 {{5.0, 2.0, 4.0},
                   {1.0, 4.0, 7.0},
                   {2.0, 3.0, 6.0}}
tm2    matrix_3x3 {{2.0, 3.0, 8.0},
                   {3.0, 3.0, 5.0},
                   {7.0, 9.0, 1.0}}
tm1new matrix_3x3  ?

lea  eax, DWORD PTR tm1new
lea  edx, DWORD PTR tm1
lea  ecx, DWORD PTR tm2
call  AMD3D_Matrix_Mul_3x3

push  DWORD PTR tm1new
call  print_matrix_3x3
```

[show results]

4.11.2 Matrix Multiplication (4x4 by 4x4)

Name

AMD3D_Matrix_Mul_4x4 - Multiplies 2 4x4 matrices

Synopsis

```
lea  eax, DWORD PTR result
lea  edx, DWORD PTR m1
lea  ecx, DWORD PTR m2
call AMD3D_Matrix_Mul_4x4
```

Description

Input Parameters

m1 - a pointer to a 4x4 dimensional matrix. Passed in the EDX register.

m2 - a pointer to a 4x4 dimensional matrix. Passed in the ECX register.

Output Parameters

result - a pointer to a 4x4 dimensional matrix holding the result of the matrix Multiplication operation. Passed in the EAX register.

Discussion

This function performs a matrix Multiplication where both matrices are 4x4 dimension. The result is also a matrix of 4x4 dimension.

Example

```
tm1    matrix_4x4 {{5.0, 2.0, 4.0, 6.0},
                   {1.0, 4.0, 7.0, 2.0},
                   {2.0, 8.0, 3.0, 1.0},
                   {4.0, 5.0, 6.0, 1.0}}
tm2    matrix_4x4 {{3.0, 1.0, 6.0, 8.0},
                   {3.0, 3.0, 8.0, 5.0},
                   {3.0, 7.0, 8.0, 9.0},
                   {0.0, 0.0, 0.0, 1.0}}
tm1new matrix_4x4  ?

lea  eax, DWORD PTR tm1new
lea  edx, DWORD PTR tm1
lea  ecx, DWORD PTR tm2
call  AMD3D_Matrix_Mul_4x4

push  DWORD PTR tm1new
call  print_matrix_4x4
```

[show results]

4.12 Matrix/Scalar addition

Name

AMD3D_Matrix_Scalar_Add_4x4 - Adds a scalar with a 4x4 matrix

Synopsis

```
lea  edx, DWORD PTR result
lea  ecx, DWORD PTR m1
movd  mm0, k
call AMD3D_Matrix_Mul_4x4
```

Description

Input Parameters

m1 - a pointer to a 4x4 dimensional matrix. Passed in the EDX register.

k - the scalar value. Passed in the mm0 register.

Output Parameters

result - a pointer to a 4x4 dimensional matrix holding the result of the matrix/scalar addition operation. Passed in the EDX register.

Discussion

This function performs a matrix/scalar addition where the matrix is 4x4 dimension. The scalar is of type float. The result is also a matrix of 4x4 dimension.

Example

```
tm1    matrix_4x4 {{5.0, 2.0, 4.0, 6.0},
                   {1.0, 4.0, 7.0, 2.0},
                   {2.0, 8.0, 3.0, 1.0},
                   {4.0, 5.0, 6.0, 1.0}}
tm1new matrix_4x4  ?
k_val  float       5.0

lea  edx, DWORD PTR tm1new
lea  edx, DWORD PTR tm1
movd mm0, k_val
call  AMD3D_Matrix_Scalar_Add_4x4

push  DWORD PTR tm1new
call  print_matrix_4x4
```

[show results]

4.13 Matrix/Scalar subtraction

Name

AMD3D_Matrix_Scalar_Sub_4x4 - Subtracts a scalar from a 4x4 matrix

Synopsis

```
lea  edx, DWORD PTR result
lea  ecx, DWORD PTR m1
movd  mm0, k
call AMD3D_Matrix_Mul_4x4
```

Description

Input Parameters

m1 - a pointer to a 4x4 dimensional matrix. Passed in the EDX register.

k - the scalar value. Passed in the mm0 register.

Output Parameters

result - a pointer to a 4x4 dimensional matrix holding the result of the matrix/scalar subtraction operation. Passed in the EDX register.

Discussion

This function performs a matrix/scalar subtraction where the matrix is 4x4 dimension. The scalar is of type float. The result is also a matrix of 4x4 dimension.

Example

```
tm1    matrix_4x4 {{5.0, 2.0, 4.0, 6.0},
                   {1.0, 4.0, 7.0, 2.0},
                   {2.0, 8.0, 3.0, 1.0},
                   {4.0, 5.0, 6.0, 1.0}}
tm1new matrix_4x4  ?
k_val  float       5.0

lea  edx, DWORD PTR tm1new
lea  edx, DWORD PTR tm1
movd mm0, k_val
call  AMD3D_Matrix_Scalar_Sub_4x4

push  DWORD PTR tm1new
call  print_matrix_4x4
```

[show results]

4.14 Matrix/Scalar multiplication

Name

AMD3D_Matrix_Scalar_Mul_4x4 - Multiplies a scalar with a 4x4 matrix

Synopsis

```
lea  edx, DWORD PTR result
lea  ecx, DWORD PTR m1
movd  mm0, k
call AMD3D_Matrix_Mul_4x4
```

Description

Input Parameters

m1 - a pointer to a 4x4 dimensional matrix. Passed in the EDX register.

k - the scalar value. Passed in the mm0 register.

Output Parameters

result - a pointer to a 4x4 dimensional matrix holding the result of the matrix/scalar multiplication operation. Passed in the EDX register.

Discussion

This function performs a matrix/scalar multiplication where the matrix is 4x4 dimension. The scalar is of type float. The result is also a matrix of 4x4 dimension.

Example

```
tm1    matrix_4x4 {{5.0, 2.0, 4.0, 6.0},
                   {1.0, 4.0, 7.0, 2.0},
                   {2.0, 8.0, 3.0, 1.0},
                   {4.0, 5.0, 6.0, 1.0}}
tm1new matrix_4x4  ?
k_val  float       5.0

lea  edx, DWORD PTR tm1new
lea  edx, DWORD PTR tm1
movd mm0, k_val
call  AMD3D_Matrix_Scalar_Mul_4x4

push  DWORD PTR tm1new
call  print_matrix_4x4
```

[show results]

4.15 Single Vector Transform

4.15.1 Vector Transform (1x4 by 4x4)

Name

AMD3D_Vector_Transform_4 - transform 1x4 vector using 4x4 matrix

Synopsis

```
lea  eax, DWORD PTR result
lea  edx, DWORD PTR mh1
lea  ecx, DWORD PTR vh1
call AMD3D_Vector_Transform_4
```

Description

Input Parameters

mh1 - a pointer to a 4x4 dimensional homogeneous matrix which is the transformation matrix. Passed in the ECX register.

vh1 - a pointer to a 1x4 dimensional homogeneous matrix holding the x, y, z and w values. Passed in the EDX register.

Output Parameters

result - a pointer to a 1x4 dimensional homogeneous matrix holding the result of the vector transform operation. Passed in the EAX register.

Discussion

This function performs a matrix multiplication of a vector_3d_h type with a matrix_4x4 type producing a resultant vector_3d_h value.

Example

```
vh1   vector_3d  {1.0, 2.0, 3.0, 1.0}
mh1   matrix_4x4 {{5.0, 2.0, 4.0, 6.0},
                  {1.0, 4.0, 7.0, 2.0},
                  {2.0, 8.0, 3.0, 1.0},
                  {0.0, 0.0, 0.0, 1.0}}
vh1new vector_3d ?

lea  eax, DWORD PTR vh1new
lea  edx, DWORD PTR mh1
lea  ecx, DWORD PTR vh1
call  AMD3D_Vector_Transform_4

push  DWORD PTR vh1new
call  print_vector

x = 13, y = 34, z = 27, w = 14
```

4.15.2 Vector Transform (4x4 by 1x4)

Name

AMD3D_Vector_Transform_4r - transform 1x4 vector using 4x4 matrix (reverse order)

Synopsis

```
lea  eax, DWORD PTR result
lea  edx, DWORD PTR mh1
lea  ecx, DWORD PTR vh1
call AMD3D_Vector_Transform_4r
```

Description

Input Parameters

mh1 - a pointer to a 4x4 dimensional homogeneous matrix which is the transformation matrix. Passed in the ECX register.

vh1 - a pointer to a 1x4 dimensional homogeneous matrix holding the x, y, z and w values. Passed in the EDX register.

Output Parameters

result - a pointer to a 1x4 dimensional homogeneous matrix holding the result of the vector transform operation. Passed in the EAX register.

Discussion

This function performs a matrix multiplication of a vector_3d_h type with a matrix_4x4 type producing a resultant vector_3d_h value. Order is reversed.

Example

```
vh1   vector_3d  {1.0, 2.0, 3.0, 1.0}
mh1   matrix_4x4 {{5.0, 2.0, 4.0, 6.0},
                  {1.0, 4.0, 7.0, 2.0},
                  {2.0, 8.0, 3.0, 1.0},
                  {0.0, 0.0, 0.0, 1.0}}
vh1new vector_3d ?

lea  eax, DWORD PTR vh1new
lea  edx, DWORD PTR mh1
lea  ecx, DWORD PTR vh1
call  AMD3D_Vector_Transform_4r

push  DWORD PTR vh1new
call  print_vector
```

[show results]

4.15.3 Vector Transform (1x3 by 3x3)

Name

AMD3D_Vector_Transform_3 - transform 1x3 vector using 3x3 matrix

Synopsis

```
lea  eax, DWORD PTR result
lea  edx, DWORD PTR mh1
lea  ecx, DWORD PTR vh1
call AMD3D_Vector_Transform_3
```

Description

Input Parameters

mh1 - a pointer to a 4x4 dimensional homogeneous matrix which is the transformation matrix. Passed in the ECX register.

vh1 - a pointer to a 1x4 dimensional homogeneous matrix holding the x, y, z and w values. Passed in the EDX register.

Output Parameters

result - a pointer to a 1x4 dimensional homogeneous matrix holding the result of the vector transform operation. Passed in the EAX register.

Discussion

This function performs a matrix multiplication of a vector_3d_h type with a matrix_4x4 type producing a resultant vector_3d_h value.

Example

```
vh1   vector_3d  {1.0, 2.0, 3.0, 1.0}
m1    matrix_3x3 {{5.0, 2.0, 4.0},
                  {1.0, 4.0, 7.0},
                  {2.0, 3.0, 6.0}}
vh1new vector_3d ?

lea  eax, DWORD PTR vh1new
lea  edx, DWORD PTR m1
lea  ecx, DWORD PTR vh1
call  AMD3D_Vector_Transform_3

push  DWORD PTR vh1new
call  print_vector
```

[show results]

4.15.4 Vector Transform (3x3 by 1x3)

Name

AMD3D_Vector_Transform_3r - transform 1x3 vector using 3x3 matrix (reverse order).

Synopsis

```
lea  eax, DWORD PTR result
lea  edx, DWORD PTR mh1
lea  ecx, DWORD PTR vh1
call AMD3D_Vector_Transform_3r
```

Description

Input Parameters

mh1 - a pointer to a 4x4 dimensional homogeneous matrix which is the transformation matrix. Passed in the ECX register.

vh1 - a pointer to a 1x4 dimensional homogeneous matrix holding the x, y, z and w values. Passed in the EDX register.

Output Parameters

result - a pointer to a 1x4 dimensional homogeneous matrix holding the result of the vector transform operation. Passed in the EAX register.

Discussion

This function performs a matrix multiplication of a vector_3d_h type with a matrix_4x4 type producing a resultant vector_3d_h value. Order is reversed.

Example

```
vh1   vector_3d  {1.0, 2.0, 3.0, 1.0}
m1    matrix_3x3 {{5.0, 2.0, 4.0},
                  {1.0, 4.0, 7.0},
                  {2.0, 3.0, 6.0}}
vh1new vector_3d ?

lea  eax, DWORD PTR vh1new
lea  edx, DWORD PTR m1
lea  ecx, DWORD PTR vh1
call  AMD3D_Vector_Transform_3r

push  DWORD PTR vh1new
call  print_vector
```

[show results]

4.16 Array Vector Transform

4.16.1 Array Vector Transform (1x4 by 4x4)

Name

AMD3D_Vector_Array_Transform_4 - perform a vector transform on an array of vectors.

Synopsis

```
mov  ecx, nvec
lea  edx, DWORD PTR in_ptr4
lea  eax, DWORD PTR m4x1
push eax
lea  eax, DWORD PTR out_ptr4
call AMD3D_Vector_Array_Transform_4
```

Description

Input Parameters

m4x1 - a pointer to a 4x4 dimensional homogeneous matrix. Passed in the ECX register.
in_ptr4 - a pointer to an array of 1x4 dimensional homogeneous matrices holding the x, y, z and w values. Passed in the EDX register.
nvec - the number of vectors in the array

Output Parameters

out_ptr4 - a pointer to an array of 1x4 dimensional homogeneous matrices holding the results of the vector array transform operation. Passed in the EAX register.

Discussion

This function performs a vector transform on an array of matrices where the matrices are vector_3d type. The results is an array of vector_3d type.

Example

```
in_ptr vector_3d_h 3 dup {{1.0, 2.0, 3.0, 1.0},
                          {5.0, 3.0, 4.0, 1.0},
                          {6.0, 8.0, 4.0, 1.0}}
mh1   matrix_4x4 {{5.0, 2.0, 4.0, 6.0},
                  {1.0, 4.0, 7.0, 2.0},
                  {2.0, 8.0, 3.0, 1.0},
                  {0.0, 0.0, 0.0, 1.0}}
out_ptr vector_3d_h 3 dup(?)
nvec   DD         3

mov  ecx, nvec
lea  edx, DWORD PTR in_ptr
lea  eax, DWORD PTR mh1
push eax
lea  eax, DWORD PTR out_ptr
call  AMD3D_Vector_Array_Transform_4

push  DWORD PTR out_ptr
push  nvec
call  print_vector_array
```

[show results]

4.16.2 Array Vector Transform (4x4 by 1x4)

Name

AMD3D_Vector_Array_Transform_4r - perform a vector transform on an array of vectors. (reverse order)

Synopsis

```
mov  ecx, nvec
lea  edx, DWORD PTR in_ptr4
lea  eax, DWORD PTR m4x1
push eax
lea  eax, DWORD PTR out_ptr4
call AMD3D_Vector_Array_Transform_4r
```

Description

Input Parameters

m4x1 - a pointer to a 4x4 dimensional homogeneous matrix. Passed in the ECX register.
in_ptr4 - a pointer to an array of 1x4 dimensional homogeneous matrices holding the x, y, z and w values. Passed in the EDX register.
nvec - the number of vectors in the array

Output Parameters

out_ptr4 - a pointer to an array of 1x4 dimensional homogeneous matrices holding the results of the vector array transform operation. Passed in the EAX register.

Discussion

This function performs a vector transform on an array of matrices where the matrices are vector_3d type. The results is an array of vector_3d type. Order is reversed.

Example

```
in_ptr vector_3d_h 3 dup {{1.0, 2.0, 3.0, 1.0},
                          {5.0, 3.0, 4.0, 1.0},
                          {6.0, 8.0, 4.0, 1.0}}
mh1   matrix_4x4 {{5.0, 2.0, 4.0, 6.0},
                  {1.0, 4.0, 7.0, 2.0},
                  {2.0, 8.0, 3.0, 1.0},
                  {0.0, 0.0, 0.0, 1.0}}
out_ptr vector_3d_h    3 dup(?)
nvec   DD         3

mov  ecx, nvec
lea  edx, DWORD PTR in_ptr
lea  eax, DWORD PTR mh1
push eax
lea  eax, DWORD PTR out_ptr
call  AMD3D_Vector_Array_Transform_4r

push  DWORD PTR out_ptr
push  nvec
call  print_vector_array

[show results]
```

4.16.3 Array Vector Transform (1x3 by 3x3)

Name

AMD3D_Vector_Array_Transform_3 - perform a vector transform on an array of vectors.

Synopsis

```
mov  ecx, nvec
lea  edx, DWORD PTR in_ptr3
lea  eax, DWORD PTR m3x1
push eax
lea  eax, DWORD PTR out_ptr3call
AMD3D_Vector_Array_Transform_3
```

Description

Input Parameters

m3x1 - a pointer to a 3x3 dimensional matrix. Passed in the ECX register.
in_ptr3 - a pointer to an array of 1x3 dimensional matrices holding the x, y, z and w values. Passed in the EDX register.
nvec - the number of vectors in the array

Output Parameters

out_ptr3 - a pointer to an array of 1x3 dimensional homogeneous matrices holding the results of the vector array transform operation. Passed in the EAX register.

Discussion

This function performs a vector transform on an array of matrices where the matrices are vector_3d type. The results is an array of vector_3d type.

Example

```
in_ptr vector_3d 3 dup {{1.0, 2.0, 3.0},
                        {5.0, 3.0, 4.0},
                        {6.0, 8.0, 4.0}}
m1    matrix_3x3 {{5.0, 2.0, 4.0},
                  {1.0, 4.0, 7.0},
                  {2.0, 3.0, 6.0}}
out_ptr vector_3d  3 dup(?)
nvec   DD         3

lea  ecx, DWORD PTR m1
lea  edx, DWORD PTR in_ptr
lea  eax, DWORD PTR out_ptr
movd mm0, nvec
call  AMD3D_Vector_Array_Transform_3

push  DWORD PTR out_ptr
push  nvec
call  print_vector_array
```

[show results]

4.16.4 Array Vector Transform (3x3 by 1x3)

Name

AMD3D_Vector_Array_Transform_3r - perform a vector transform on an array of vectors. (reverse order)

Synopsis

```
mov  ecx, nvec
lea  edx, DWORD PTR in_ptr3
lea  eax, DWORD PTR m3x1
push eax
lea  eax, DWORD PTR out_ptr3call
AMD3D_Vector_Array_Transform_3r
```

Description

Input Parameters

m3x1 - a pointer to a 4x4 dimensional homogeneous matrix. Passed in the ECX register.
in_ptr3 - a pointer to an array of 1x3 dimensional matrices holding the x, y, and z values. Passed in the EDX register.
nvec - the number of vectors in the array

Output Parameters

out_ptr3 - a pointer to an array of 1x3 dimensional matrices holding the results of the vector array transform operation. Passed in the EAX register.

Discussion

This function performs a vector transform on an array of matrices where the matrices are vector_3d type. The results is an array of vector_3d type. Order is reversed.

Example

```
in_ptr vector_3d 3 dup {{1.0, 2.0, 3.0},
                        {5.0, 3.0, 4.0},
                        {6.0, 8.0, 4.0}}
m1    matrix_3x3 {{5.0, 2.0, 4.0},
                  {1.0, 4.0, 7.0},
                  {2.0, 3.0, 6.0}}
out_ptr vector 3d  3 dup(?)
nvec   DD         3

mov  ecx, nvec
lea  edx, DWORD PTR in_ptr
lea  eax, DWORD PTR m3x1
push eax
lea  eax, DWORD PTR out_ptr3call
AMD3D_Vector_Array_Transform_3r

push  DWORD PTR out_ptr
push  nvec
call  print_vector_array
```

[show results]

4.17 Dot Product

Name

AMD3D_Vector_Dot_Product - calculate the Dot Product of 2 1x4 vectors

Synopsis

```
lea ecx, dword ptr v1
lea edx, dword ptr v2
lea eax, dword ptr result
call AMD3D_Vector_Dot_Product3
```

Description

Input Parameters

v1 - a pointer to a 1x4 dimensional homogeneous matrix holding the x, y, z and w values. Passed in the ECX register.

v2 - a pointer to a 1x4 dimensional homogeneous matrix holding the x, y, z and w values. Passed in the EDX register.

Output Parameters

result - a pointer to a float holding the result of the vector dot product operation. Passed in the EAX register.

Discussion

This function performs a vector dot product on 2 matrices where the matrices are vector_3d type. The result is a float type.

Example

```
v1    vector_3d_h {1.0, 2.0, 3.0, 1.0}
v2    vector_3d_h {3.0, 5.0, 6.0, 1.0}
result float      ?

lea ecx, dword ptr v1
lea edx, dword ptr v2
lea eax, dword ptr result
call  AMD3D_Vector_Dot_Product

push  DWORD PTR result
call  print_float
```

[show results]

4.18 Cross Product

Name

AMD3D_Vector_Cross_Product - calculate the Cross Product of 2 1x4 vectors

Synopsis

```
lea  ecx, DWORD PTR vh1
lea  edx, DWORD PTR vh2
lea  eax, DWORD PTR result
call AMD3D_Vector_Cross_Product
```

Description

Input Parameters

vh1 - a pointer to a 1x4 dimensional homogeneous matrix holding the x, y, z and w values. Passed in the ECX register.

vh2 - a pointer to a 1x4 dimensional homogeneous matrix holding the x, y, z and w values. Passed in the EDX register.

Output Parameters

result - a pointer to a 1x4 dimensional homogeneous matrix holding the resulting x, y, z and w values from the cross product. Passed in the EAX register.

Discussion

This function performs a vector cross product on 2 matrices where the matrices are vector_3d type. The result is a vector_3d type.

Example

```
vh1    vector_3d {1.0, 2.0, 3.0, 1.0}
vh2    vector_3d {3.0, 5.0, 6.0, 1.0}
v1new  vector_3d ?

lea  ecx, DWORD PTR vh1
lea  edx, DWORD PTR vh1
lea  eax, DWORD PTR v1new
call  AMD3D_Vector_Cross_Product

push  DWORD PTR v1new
call  print_vector
```

[show results]

4.19 Array dot products3

Name

AMD3D_Vector_Array_Dot_Product3 - calculate the Dot Product of 1 1x3 vector with an array of 1x3 vectors

Synopsis

```
mov  ecx, nvec
lea  edx, DWORD PTR in_ptr3
lea  eax, DWORD PTR vnh1
push eax
lea  eax, DWORD PTR result
call AMD3D_Vector_Array_Dot_Product3
```

Description

Input Parameters

vnh1 - a pointer to a 1x3 dimensional homogeneous matrix holding the x, y, z values. Passed on the stack.

in_ptr3 - a pointer to an array of 1x4 dimensional homogeneous matrices holding the x, y, z values. Passed in the EDX register.

nvec - the number of vectors in the array

Output Parameters

result - a pointer to an array of floats holding the result of the vector dot product operation. Passed in the EAX register.

Discussion

This function performs a vector dot product on an array of matrices where the matrices are vector_3d type. The results is an array of float type.

Example

```
in_ptr vector_3d 3 dup {{1.0, 2.0, 3.0, 1.0},
                        {5.0, 3.0, 4.0, 1.0},
                        {6.0, 8.0, 4.0, 1.0}}
v1 vector_3d {3.0, 5.0, 6.0, 1.0}
result float      3 dup(?)
nvec   DD         3

mov  ecx, nvec
lea  edx, DWORD PTR in_ptr3
lea  eax, DWORD PTR v1
push eax
lea  eax, DWORD PTR result
call  AMD3D_Vector_Array_Dot_Product3

push  DWORD PTR result
push  nvec
call  print_float_array
```

[show results]

4.20 Array dot products4

Name

AMD3D_Vector_Array_Dot_Product4 - calculate the Dot Product of 1 1x4 vector with an array of 1x4 vectors

Synopsis

```
mov  ecx, nvec
lea  edx, DWORD PTR in_ptr4
lea  eax, DWORD PTR v1
push eax
lea  eax, DWORD PTR result
call AMD3D_Vector_Array_Dot_Product4
```

Description

Input Parameters

v1 - a pointer to a 1x4 dimensional homogeneous matrix holding the x, y, z and w values. Passed on the stack.

in_ptr4 - a pointer to an array of 1x4 dimensional homogeneous matrices holding the x, y, z and w values. Passed in the EDX register.

nvec - the number of vectors in the array. Passed in the ECX register.

Output Parameters

out_ptr4 - a pointer to an array of floats holding the result of the vector dot product operation. Passed in the EAX register.

Discussion

This function performs a vector dot product on an array of matrices where the matrices are vector_3d_h type. The results is an array of float type.

Example

```
in_ptr vector_3d_h 3 dup {{1.0, 2.0, 3.0, 1.0},
                          {5.0, 3.0, 4.0, 1.0},
                          {6.0, 8.0, 4.0, 1.0}}
vh1 vector_3d_h {3.0, 5.0, 6.0, 1.0}
result float      3 dup(?)
nvec   DD         3

mov  ecx, nvec
lea  edx, DWORD PTR in_ptr
lea  eax, DWORD PTR vh1
push eax
lea  eax, DWORD PTR result
call  AMD3D_Vector_Array_Dot_Product4

push  DWORD PTR result
push  nvec
call  print_float_array
```

[show results]

4.21 Array cross products

Name

AMD3D_Vector_Array_Cross_Product - calculate the Cross Product of 1 1x3 vector with an array of vectors

Synopsis

```
movd ecx, nvec
lea  edx, DWORD PTR in_ptr3
lea  eax, DWORD PTR vnh1
push eax
lea  eax, DWORD PTR out_ptr3
call AMD3D_Vector_Array_Cross_Product
```

Description

Input Parameters

vnh1 - a pointer to a 1x3 dimensional homogeneous matrix holding the x, y, z values. Passed on the stack.

in_ptr3 - a pointer to an array of 1x3 dimensional homogeneous matrices holding the x, y, z values. Passed in the EDX register.

nvec - the number of vectors in the array. Passed in the ECX register.

Output Parameters

out_ptr3 - a pointer to an array of 1x3 dimensional homogeneous matrix holding the x, y, z values which are the results of the vector cross product operation. Passed in the EAX register.

Discussion

This function performs a vector cross product on an array of matrices where the matrices are vector_3d type. The results is an array of vector_3d types.

Example

```
in_ptr vector_3d 3 dup {{1.0, 2.0, 3.0},
                        {5.0, 3.0, 4.0},
                        {6.0, 8.0, 4.0}}
v1 vector_3d {3.0, 5.0, 6.0}
result vector_3d  3 dup(?)
nvec   DD         3

movd ecx, nvec
lea  edx, DWORD PTR in_ptr
lea  eax, DWORD PTR v1
push eax
lea  eax, DWORD PTR out_ptr3
call  AMD3D_Vector_Array_Cross_Product

push  DWORD PTR result
push  nvec
call  print_vector_array
```

[show results]

5. Transcendentals

5.1 sine

Name

AMD3D_sin - perform a sine function on a value

Synopsis

```
movd mm0, angle
call AMD3D_sin
movd result, mm0
```

Description

Input Parameters

angle - the angle expressed in radians

Output Parameters

result - the sine of the angle

Discussion

This function performs a sine function on a value expressed in radians. The angle is passed in mm0 and the result is returned in mm0.

Example

```
angle      float 10.0
sin_angle  float ?

movd mm0, angle
call AMD3D_sin
movd sin_angle, mm0

push  sin_angle
call  print_float
```

[show results]

5.2 cosine

Name

AMD3D_cos - perform a cosine function on a value

Synopsis

```
movd mm0, angle
call AMD3D_cos
movd result, mm0
```

Description

Input Parameters

angle - the angle expressed in radians

Output Parameters

result - the cosine of the angle

Discussion

This function performs a cosine function on a value expressed in radians. The angle is passed in mm0 and the result is returned in mm0.

Example

```
angle       float 10.0
cos_angle   float ?

movd mm0, angle
call AMD3D_cos
movd cos_angle, mm0

push  cos_angle
call  print_float
```

[show results]

5.3 combined sine and cosine

Name

AMD3D_sincos - perform a sine and cosine function on a value

Synopsis

```
movd mm0, angle
call AMD3D_sincos
movq result[0], mm0
```

Description

Input Parameters

angle - the angle expressed in radians

Output Parameters

result[0],result[1] - the cosine and sine of the angle respectively.

Discussion

This function performs a simultaneous sine and cosine function on a value expressed in radians. The angle is passed in mm0 and the result is returned in mm0. Lower half is cosine and upper half is sine.

Example

```
angle        float    10.0
cosin_angle  float    ?
sin_angle    float    ?

movd mm0, angle
call AMD3D_cosine
movq cosin_angle, mm0

push  cosin_angle
call  print_float

push  sin_angle
call  print_float
```

[show results]

5.4 arctangent

Name

AMD3D_atan - perform an arctangent function on a value

Synopsis

```
movd mm0, angle
call AMD3D_atan
movq result, mm0
```

Description

Input Parameters

angle - the angle expressed in radians

Output Parameters

result - the arctangent of the angle

Discussion

This function performs an arctangent function on a value expressed in radians. The angle is passed in mm0 and the result is returned in mm0.

Example

```
angle         float   10.0
arctan_angle  float   ?

movd mm0, angle
call AMD3D_atan
movq arctan_angle, mm0

push  arctan_angle
call  print_float
```

[show results]

5.5 natural logarithm

Name

AMD3D_natlog - perform a natural logarithm function on a value

Synopsis

```
movd mm0, value
call AMD3D_log
movq result, mm0
```

Description

Input Parameters

value - the input value

Output Parameters

result - the natural logarithm of the value

Discussion

This function performs a natural logarithm function on a value. The value is passed in mm0 and the result is returned in mm0.

Example

```
x          float      5.34
ln_x       float      ?

movd mm0, x
call AMD3D_log
movq ln_x, mm0

push  ln_x
call  print_float
```

[show results]

6. Image Processing

6.1 Forward DCT (Discrete Cosine Transform)

Name

AMD3D_JPEG_FDCT - JPEG style Forward Discrete Cosine Transform

Synopsis

```
void __fastcall AMD3D_JPEG_FDCT(float *dp);
```

Description

Input Parameters

dp - a pointer to an array of float values

Output Parameters

dp - a pointer to an array of float values

Discussion

This function performs a forward discrete cosine transform in the style used in JPEG compression. The transform is computed in place so input values are overwritten. The DCT size is fixed at 8 for this algorithm so the area processed is 8 by 8.

Example

```
float d[8][8] = {1.0F,2.0F,3.0F,4.0F,5.0F,6.0F,7.0F,8.0F,
                 2.0F,2.0F,3.0F,4.0F,5.0F,6.0F,7.0F,8.0F,
                 3.0F,2.0F,3.0F,4.0F,5.0F,6.0F,7.0F,8.0F,
                 4.0F,2.0F,3.0F,4.0F,5.0F,6.0F,7.0F,8.0F,
                 5.0F,2.0F,3.0F,4.0F,5.0F,6.0F,7.0F,8.0F,
                 6.0F,2.0F,3.0F,4.0F,5.0F,6.0F,7.0F,8.0F,
                 7.0F,2.0F,3.0F,4.0F,5.0F,6.0F,7.0F,8.0F,
                 8.0F,2.0F,3.0F,4.0F,5.0F,6.0F,7.0F,8.0F};

print_matrix_8x8(&d);
AMD3D_JPEG_FDCT(&d);
print_matrix_8x8(&d);
```

7. Macros

7.1 Macros for inline assembly in C

7.1.1 AMD 3D instructions

Insert AMD3DA.H macros here.

7.1.2 Simple functions

These macros are provided in a form usable in __asm statements within C files. You must include AMD3DB.H in any file in which they are used.

7.1.2.1 square root macro

Name
AMD3D_square_root_cmac - perform a square root function on a value
Synopsis

```
AMD3D_square_root_cmac(destination, source)
```

Description
Input Parameters
source - the input value
Output Parameters
destination - the square root of the input value
Discussion
This macro inserts AMD 3D code which performs a square root function on a value. The value is specified as the source operand and the result is specified as the destination operand.
Example

```
__ASM {

      AMD3D_square_root_cmac(mm0, mm1)
}
```

7.1.2.2 square macro

Name

AMD3D_square_cmac - perform a square function on a value

Synopsis

```
AMD3D_square_cmac(destination, source)
```

Description

Input Parameters

source - the input value

Output Parameters

destination - the square root of the input value

Discussion

This macro inserts AMD 3D code which performs a square function on a value. The value is specified as the source operand and the result is specified as the destination operand.

Example

```
__ASM {

      AMD3D_square_cmac(mm0,mm1)
}
```

7.1.2.3 reciprocal square root macro

Name

AMD3D_recip_square_root_cmac - perform a reciprocal square root function on a value

Synopsis

```
AMD3D_recip_square_root_cmac(destination, source)
```

Description

Input Parameters

source - the input value

Output Parameters

destination - the reciprocal square root of the input value

Discussion

This macro inserts AMD 3D code which performs a reciprocal square root function on a value. The value is specified as the source operand and the result is specified as the destination operand.

Example

```
__ASM {

      AMD3D_recip_square_root_cmac(mm1,mm0)
}
```

7.1.2.4 divide macro

Name

AMD3D_divide_cmac - perform a floating point divide operation with 2 values

Synopsis

```
AMD3D_divide_cmac(destination, source)
```

Description

Input Parameters

source - the divisor for the divide operation
destination - the dividend for the divide operation

Output Parameters

destination - the result of the divide

Discussion

This macro performs a divide operation with 2 values. The dividend is specified as the destination operand, the divisor is specified as the source operand and the result is also specified with the destination operand.

Example

```
__ASM {

      AMD3D_divide_cmac(mm0, mm1)
}
```

7.1.2.5 Integer to float conversion macro

Name

AMD3D_ltof_cmac - perform a integer to floating point conversion on a value

Synopsis

```
AMD3D_ltof_cmac(destination, source)
```

Description

Input Parameters

source - the integer value

Output Parameters

destination - the floating point equivalent

Discussion

This macro performs an integer to floating point conversion on a value. The value is specified as the source operand and the result is specified as the destination operand.

Example

```
__ASM {

      AMD3D_ltof_cmac(mm0, mm1)
}
```

7.1.2.6 Float to integer conversion macro

Name

AMD3D_ftol_cmac - perform a floating point to integer conversion on a value

Synopsis

```
AMD3D_ftol_cmac(destination, source)
```

Description

Input Parameters

source - the floating point value

Output Parameters

destination - the integer equivalent

Discussion

This macro performs an integer to floating point conversion on a value. The value is specified as the source operand and the result is specified as the destination operand.

Example

```
__ASM {

      AMD3D_ftol_cmac(mm0, mm1)
}
```

7.1.2.7 FLD constant macros

Name

AMD3D_FLDxx_cmac - perform a floating point constant load to register

Synopsis

```
AMD3D_FLDxx_cmac(destination)    where xx is:
1 - one
L2T - log2(10)
L2E - Log2(e)
PI - PI
LG2 - Log10(2)
LN2 - Loge(2)
Z - load zero
```

Description

Input Parameters

Output Parameters

destination - the register which the constant is loaded into.

Discussion

These macros performs a Constant load into an MMX register. The following types are supported:

AMD3D_FLDLN2_cmac
AMD3D_FLDLG2_cmac
AMD3D_FLDPI_cmac
AMD3D_FLDL2E_cmac
AMD3D_FLDL2T_cmac
AMD3D_FLD1_cmac
AMD3D_FLDZ_cmac

Example

```
__ASM {

      AMD3D_FLDZ_cmac(mm0, mm1)
}
```

7.2 Macros for assembly code

These macros are created for frequently used operations and are more general than the function call equivalents above.

7.2.1 Simple operations

These operations are provided as an alternative to the inline assembly approach above. It is assumed that these functions are called from other AMD-3D assembly. Also these functions are more flexible in that the source and destination are not fixed to registers only. You must include AMD3DMAC.INC file in any file in which they are used.

7.2.1.1 square root macro

Name

AMD3D_square_root_mac - perform a square root function on a value

Synopsis

```
AMD3D_square_root_mac destination, source
```

Description

Input Parameters

source - the input value

Output Parameters

destination - the square root of the input value

Discussion

This macro inserts AMD 3D code which performs a square root function on a value. The value is specified as the source operand and the result is specified as the destination operand.

Example

```
value      float      2.0
sq_rt_val  float      ?

AMD3D_square_root_mac  sq_rt_val, value

push  sq_rt_val
call  print_float
```

[show results]

7.2.1.2 square macro

Name

AMD3D_square_mac - perform a square function on a value

Synopsis

```
AMD3D_square_mac destination, source
```

Description

Input Parameters

source - the input value

Output Parameters

destination - the square root of the input value

Discussion

This macro inserts AMD 3D code which performs a square function on a value. The value is specified as the source operand and the result is specified as the destination operand.

Example

```
value      float      2.0
sq_val     float      ?

AMD3D_square_mac sq_val, value

push  sq_val
call  print_float
```

[show results]

7.2.1.3 reciprocal square root macro

Name

AMD3D_recip_square_root_mac - perform a reciprocal square root function on a value

Synopsis

```
AMD3D_recip_square_root_mac  destination, source
```

Description

Input Parameters

source - the input value

Output Parameters

destination - the reciprocal square root of the input value

Discussion

This macro inserts AMD 3D code which performs a reciprocal square root function on a value. The value is specified as the source operand and the result is specified as the destination operand.

Example

```
value           float      2.0
recip_sq_rt_val float      ?

AMD3D_recip_square_root_mac  recip_sq_rt_val, value

push  recip_sq_rt_val
call  print_float
```

[show results]

7.2.1.4 divide macro

Name

AMD3D_divide_mac - perform a floating point divide operation with 2 values

Synopsis

```
AMD3D_divide_mac destination, source
```

Description

Input Parameters

source - the divisor for the divide operation
destination - the dividend for the divide operation

Output Parameters

destination - the result of the divide

Discussion

This macro performs a divide operation with 2 values. The dividend is specified as the destination operand, the divisor is specified as the source operand and the result is also specified with the destination operand.

Example

```
dividend   float      2.0
divisor    float      7.0

AMD3D_divide_mac dividend, divisor

push  dividend
call  print_float
```

[show results]

7.2.1.5 Integer to float conversion macro

Name

AMD3D_ltof_mac - perform a integer to floating point conversion on a value

Synopsis

```
AMD3D_ltof_mac destination, source
```

Description

Input Parameters

source - the floating point value

Output Parameters

destination - the integer equivalent

Discussion

This macro performs an integer to floating point conversion on a value. The value is specified as the source operand and the result is specified as the destination operand.

Example

```
int_val    DD         3
float_val  float      ?

AMD3D_ltof_mac float_val, int_val

push  float_val
call  print_float
```

[show results]

7.2.1.6 Float to integer conversion macro

Name

AMD3D_ftol_mac - perform a floating point to integer conversion on a value

Synopsis

```
AMD3D_ftol_mac destination, source
```

Description

Input Parameters

source - the integer value

Output Parameters

destination - the floating point equivalent

Discussion

This macro performs an integer to floating point conversion on a value. The value is specified as the source operand and the result is specified as the destination operand.

Example

```
float_val  float      4.0
int_val    DD         ?

AMD3D_ftol_mac int_val, float_val

push  int_val
call  print_int
```

[show results]

7.2.1.7 Absolute value macro

Name

AMD3D_fabs_mac - perform an absolute value operation on a value

Synopsis

```
AMD3D_fabs_mac value
```

Description

Input Parameters

value - the positive or negative floating point value.

Output Parameters

value - the input value if it was positive else the negation of input value.

Discussion

This macro performs an absolute value operation on a value. The value modified in place.

Example

```
float_val  float      -4.0


AMD3D_fabs_mac int_val

push  int_val
call  print_int
```

[show results]

7.2.1.8 Change sign macro

Name

AMD3D_fchs_mac - inverts the sign of the floating point value

Synopsis

```
AMD3D_fchs_mac value
```

Description

Input Parameters

value - the positive or negative floating point value.

Output Parameters

value - the input value with: sign bit ← NOT(sign bit).

Discussion

This macro performs a toggle of the sign bit of the floating point value. The value modified in place.

Example

```
float_val  float      4.0


AMD3D_fchs_mac int_val

push  int_val
call  print_int
```

[show results]

7.2.1.9 Load constant macros

Name

AMD3D_fldxx_mac - load constant into MMX floating point register

Synopsis

```
AMD3D_fldXX_mac register - where XX is:
1 - one
L2T - log2(10)
L2E - Log2(e)
PI - PI
LG2 - Log10(2)
LN2 - Loge(2)
Z - load zero
```

Description

Input Parameters

register - the MMX register to load the constant into.

Output Parameters

register - constains the constant.

Discussion

These macros load a constant into a designated MMX register. They use the EAX register in the process. The following versions are supported:

AMD3D_FLDLN2_mac
AMD3D_FLDLG2_mac
AMD3D_FLDPI_mac
AMD3D_FLDL2E_mac
AMD3D_FLDL2T_mac
AMD3D_FLD1_mac
AMD3D_FLDZ_mac

Example

```
const float ?

AMD3D_fldL2E_mac mm0
movd const, mm0

push  const
call  print_float
```

[show results]